P390/Linux
User's Guide
Version 1 Release 2

Document Number jh00-0101-00

June 9, 2005 11:42 a.m.

John P. Hartmann

Draft—DO NOT PUBLISH

Second Edition, May 2005.

Contents

Preface

Intended Audience

This book is written for systems programmers who run a P390/Linux installation. Basic UNIX skills are assumed, as is a general knowledge of IBM Enterprise System/390.

Syntax Diagrams

Syntax for MANOPS subcommands and the device map file is shown in standard “tracks” diagrams.

In the diagrams, keywords are shown with the minimum abbreviation in uppercase and the remaining optional part in lowercase. Case is ignored when matching keywords.

Syntactic variables are represented by words beginning with a lowercase letter. Case is respected in character variables, such as words (blank-delimited), delimited strings (/abc/) and strings (to the end of the line).

Command Switches

The commands accept standard unix switches, which are prefixed a hyphen, as well as positional operands. A switch may represent a binary value or it may be followed by an operand. Case is respected both in the switches (which are all in lowercase) and their operands. Some switches have a one-character abbreviation.

Examples of switches are `--verbose`, which produces additional output; it may be abbreviated to `-v`. An example of a switch taking an operand is `--devmap p390.devmap`. Unlike standard unix, the switches are words; binary switches cannot be aggregated.

Terminology

A P390 system consists of hardware and supporting software.

Device driver	A kernel module that interfaces to the P390 card.
Subchannel file	A special file created in `/dev/p390` to represent a P390 subchannel. The interface is like the old bus and tag lines.
Device manager	A user-mode program that interfaces a P390 special character subchannel file to a Linux file or other interface. That is, the device manager maps the subchannel file to a Linux resource.
`manops`	Manual Operations; the service processor. The user interface to the emulated System/390 hardware.
Started task	A program that runs as a child of the `manops` process, but does not handle a subchannel file.

Acknowledgements

The compression routines used for the CKE_P370 file format and for the jphformat tapes are from zlib 1.1.3 (C) 1995-1998 Jean-loup Gailly and Mark Adler. To simplify installation, the object code has been linked into the 3390 device manager. I used the distribution dated 9 July 1998 without modification. I appreciate the authors' largresse in allowing any use of their code whatsoever.

Overview

A P390/Linux system implements IBM Enterprise System/390 at Generation 3. It consists of a PCI card and Linux software running on a 32-bit Intel Pentium processor.

The P390/Linux code that runs in the kernel is often called the device driver. It has access to the PCI bus and can control the P390 card. It implements the channel functions up to the channel/control unit interface as a virtual file system. These special device files access the P390 subchannels. Another special file allows access to the manual controls of the P390.

The entire XA I/O subsystem is implemented in Linux code. The cannel subsystem up to the bus and tag lines is resident in the kernel; device emulation is performed by user processes. Likewise, the manual operations interface runs in a user process.

The P390/Linux support is a complete re-implementation, but it uses many concepts from the standard R390. Unlike R390, it provides a command line interface only. The device map is a flat ASCII file that is maintained with the administrator's favourite editor.

Hardware

The P390 card implements the System/390 CPU, real storage, and subchannels, but no Input/Output hardware. The card is bus mastering; it can access emulated expanded storage directly from a block of reserved RAM.

Initalisation

Device Driver and Special Files

The kernel module that contains the PCI card driver and channel subsystem must first be loaded. As part of this initialisation, the device driver scans the PCI bus for P390 cards and resets any it finds.

For each P390 card, the device driver, loads the bootstrap microcode and lets it run to reset the device. It then creates a device major node for the card. The first major node is named `p390`; the second i named `p391` and so on up to a maximum of ten. These names are "hardwired".

If specified by parameters on the `insmod` command, ethernet interfaces are also defined.

Having installed the device driver, the initialisation shell script continues by scanning the file `/proc/devices` for device major nodes with names that begin "`p39`".

For each major node, the shell script makes a directory in the `/dev` directory corresponding to the major node name: `/dev/p390` for the first card.

The script then scans the corresponding device map file and defines special files in the directory corresponding to the major node. (`/dev/p390` directory for the first card found.) In this process, minor numbers are assigned starting 0. The minor number is also the subchannel number. The file name is the device number, for example

`/dev/p390/000c` might represent a card reader on the first P390 card. These files are made available to the user and group that have the same name as the major name.

Finally, it creates the file `manops` as subchannel 255. This file is accessible only to the user that has the same name as the major name.

If network interfaces are defined, they are configured with as point-to-point and assigned an IP address. These network interfaces represent IP stacks running in 390 operating systems.

This concludes device driver initialisation. At this time the P390(s) are reset, but have no operational microcode loaded.

The initialisation described above requires root privileges; it is normally performed as part of bringing the Linux system up in multi-user mode (runlevel 2).

`manops`

The command `manops` loads the operational microcode if it is not already loaded (from a previous execution of the `manops` command) and then performs a system reset. It then reads the device map and enables the subchannels for the specified devices.

`manops` initialises device managers that have been specified to run under `manops`' control as well as started tasks (device managers that use no subchannel file) that are specified in the device map. &devmgrs and started tasks run in separate processes.

Such a device manager is initialised with an anonymous pipe (FIFO) connected from `manops` to the device manager's standard input.

`manops` then reads commands from its standard input. These commands implement standard console functions such as display, store, reset, and IPL.

Device Manager

A typical device manager controls a single subchannel. It opens the subchannel file representing the subchannel and any &media.s that represent the emulated device medium, such as a tape reel or a disk pack.

The device manager then enters a loop to process channel operations from the subchannel file and commands from standard input.

A device manager can run in a separate session. The user must be a member of the `p390` group.

Supported Operating Systems

P390/Linux has been verified with VM/ESA 2.4 and zVM 3.1. Because the P390 processor implements only the G3 level set, zVM version 4 and later versions are not supported. zVM 4.2 and later version/releases will refuse to IPL on the hardware.

Emulated Device Types

There are device managers for the following types of devices:

1403 Line printer writes an output file with ASA carriage control. Data are translated from EBCDIC to ASCII.

3088 Channel-to-channel adapter. The adapter is supported in two variants:

1. A standard ESCON-type CTC. This requires two co-operating device managers, possibly associated with different cards.
2. A modified CTC that creates a TCP/IP link to the Linux network layer; it cannot be coupled to another subchannel. To Linux, this device manager looks like an ethernet card.

3215 The console printer/keyboard is provided to allow IPL of a P390/Linux distribution.

3270 Local 3270 devices are created by the tn3270 server.

3370 FBA driver. Supports model 1 only.

3390 ECKD driver. Irrespective of its geometry, it presents itself as a model 2. Any number of cylinders is supported, in theory up to the model-9 limit, but this will not work well with a 2G limit on individual files. The device manager supports a compressed media format, which reduces the size of the two VM packs to a tenth.

3423 Tape drive. The name refers to the optical media attach of bygone days.

3505 Card reader that reads fixed 80 files, such as downloaded IPL decks. To be able to IPL binary decks, the 3505 device manager does not convert between ASCII and EBCDIC.

Supported Operating Environments

This chapter describes the hardware and software requirements for installation and operation of P390/Linux.

Prerequisites

Hardware

The following EC levels of P390 and P390E have been tested:

```
P390 Version b, part#  08J5940 oscillator   78.375.  VPD OK in second attempt.
P390 card index 0 initialised.  Major node 254 is p390.  IRQ17.  Data at ec90d000.
P390 Version c, part#  11J4298 oscillator   25.000.
P390 card index 1 initialised.  Major node 253 is p391.  IRQ22.  Data at ecd98000.
```

Software

The p390 has been tested with these products. Other equivalent products may also work.

- SuSE Linux 6.3 with a 2.4.4 kernel. SuSE Linux 7.2 with 2.4.4, 2.4.13, 2.4.20, and 2.4.22 (Alan Cox) kernels. If your distribution is also SuSE, 7.1 is the minimum recommended level. Note that neither 2.2 nor 2.6 kernels are supported.
- PCOMM 3270 4.2 (19970822). The client *must* support the TN3270E protocol; the sever will not negotiate old-style.
- Object REXX version “OBJREXX 6.00 11 May 2000”.

Restrictions

1. zOS (MVS) has not been tested. It is likely to cause breakage in the 3390 device driver.
2. Linux/390. does not IPL because it does not find a 3390 it likes. As it requires G5 hardware, it is unlikely it would run.

Bugs

There is no doubt an abundance of unknown bugs, but the following has been observed.

- The CTC/Ethernet driver does not reconnect properly after a `SHUTDOWN REIPL` command. (November 2003.)

Caveat Emptor

Warning:

While this code has been running my P390s for some years, it may not work in all environments; and you may be putting a different load on it that will expose bugs in device managers.

The data are yours. I strongly recommend these precautions:

- Use pinned files to protect your disk images. Increase the number of pinned files when you have made significant changes to your minidisks.
- Unload your SFS to tape regularly.
- If possible, put your tape library (`/var/p390/tapes`) on a different physical volume from your disk images (`/var/p390/drives`).
- Take back-ups off-site.

PCI Hardware Installation

It is unlikely that you will want to run the P390 in the original OS/2 hardware on a 75MHz pentium. These notes are intended to assist you in moving the card to a modern system.

PCI Bus Requirement

Unless your card is one of the early P390 “development” cards, your PCI bus must supply 3.3V for the storage chips. A PCI bus that is compliant with the 2.1 specification should work; this should include all recent motherboards.

Card Installation and Verification

Any PCI slot will do as long as you have only one card. Having installed the card, boot the diagnostic diskette and run a complete diagnostic run. Do not skip this step; you will regret it later.

If the test fails in the memory test, you are likely not having 3.3V on the PCI bus. You will need to try another motherboard.

Installation

Basic Installation Procedure

These instructions assume that you have a single P390 card. Otherwise refer to Appendix A, “Using Multiple P390 Cards in one System” on page 54.

The P390/Linux support is available as a compressed tar file `p390.tar.Z`.

1. Create a `p390` user and group.
2. Edit the p390 `.profile` to include

   ```
   export LD_LIBRARY_PATH=~/lib
   export NLSPATH=~/lib/%N
   ```

   If you will be using more than the default `vm` network interface, you might wish to export the parameters for loading the device driver module, for example:

   ```
   export P390_MODULE_PARAMETERS="net=vm,lnx"
   ```

   If your distribution does not add `~/bin` to the `PATH` environment variable, make this happen.
3. Untar the distribution file in p390’s home directory. This creates `bin` and `lib` directories if they do not already exist and populates them with the device manager executables and libraries.

   Three files are put into the home directory:

   - `sp390.devmap`—a sample device map.
   - `p390.pdf`—documentation (this file).
   - `kernel.tar.Z`—the device driver module.
4. Prepare the device driver module. See “Readying the Kernel Module” on page 8.
5. Create a directory structure for the device manager files. I use `/var/p390/drives` for the disk images, `/var/p390/tapes` for the tape library, `/var/p390/printers` for print files, `/var/p390/trace` for trace files, and `/var/p390/work` for working directories files, but there is nothing that requires use of the `/var` filesystem.
6. Obtain the operating system disk images. Be sure to have a licence for the operating system. For zVM, the operating system volumes will likely come as zipped files on the integrated application system distribution CD.
7. Convert the operating system disk images to CKE_P370 format.

   Make `/var/p390/drives` the current directory. If the second Application System CD is mounted at `/cd`:

   ```
   awscvt /cd/vmesa/240w01_1.zip -batch 1000 -verify
   ```

   Verify that the converted file is correct:

```
awscke -dumpfile -in 240w01.cke
2001/07/03 08:25:33.068 awscke (c) John P. Hartmann 2001.
2001/07/03 08:25:33.069 CKE_P370 data file 0 240w01.cke fd 3.
2001/07/03 08:25:33.069 Cache allocated for 1 tracks.
File index 0 name 240w01.cke format CKE_P370:
   15 Heads, tracksize 56832 (0xde00), last cylinder 0.
   Format CKE_P370 version 1; compression type comp_zlib.
   3339 cylinders, 50085 tracks total, allocation blok size 512.
   Top index 1; 4 index blocks
   File size 423203 blocks, first free space block 0.
   (Based on st_size:  file contains 423203 blocks, remainder 0).
   File is: readonly jformat.
   Of 392 index blocks:  0 zero indexes; 392 nonzero.
   15184 tracks have data (total 422806 blocks).
   0 blank, 23144 CP zeros, 11757 CMS zeros.
   0 zero coreindex pointers where trackindex nonzero.
   Largest track is 97 blocks.
   No fragmentation in file.
   Block usages:
   0 unaccounted, 1 awsckdx, 4 topindex, 392 trackindex, 0 control, 0
2001/07/03 08:25:33.117 awscke complete return code 0.
```

While this step is not strictly required as the 3390 device manager still supports the original file format, you will reduce the file size by an order of magnitude and performance may also improve.

8. Create empty disk volumes, if additional disks are required (they are). Here, I create a 1000 cylinder disk for SPOOL and a full model 3 for the SFSPOOL1 file pool. You will probably also want a paging volume, but it is simpler to define dedicated paging packs in the device map as part of the start-up sequence.

   ```
   awscke -create -out 240s01 -cylcount 1000 -spool -label 240s01
   awscke -create -out 240w02 -model 3 -label 240w02
   ```

9. Edit the sample device map to suit your preferences and data files and save it as `p390.devmap`.

10. Install the XA microcode file AWS39X0.MCD or AWS39X1.MCD (or both!) in the home directory of the p390 user. Note that the file names are uppercase. You can find file in the root directory of the P390 diagnostics diskette.

11. Consider symbolic links from a directory in `root`'s PATH to the shell scripts that add devices dynamically:

    ```
    #/root/bin ln -s  p390/bin/addp390dev .
    #/root/bin ln -s  p390/bin/mknodp390 .
    ```

    You may also wish to link in `loadp390` and its alias names, but that is not strictly required as this command script is self-contained; it can be run by direct reference.

Readying the Kernel Module

The P390 device driver is implemented as a loadable module; it cannot be compiled into the kernel.

The kernel module is structured such that all Linux and kernel dependencies are in `p390main.c`, which you compile for your particular version of the kernel, whereas `p390x.o` contains no code related to Linux, but rather code that IBM does not wish to publish.

The kernel module is in the file `kernel.tar.Z`. It contains the tree `modules/current`; all files are in the lower directory.

You may wish to move to your kernel source path and untar the tarball there; there is no dependency in MANOPS on the location of the source and object for the the kernel module.

1. Untar the kernel module files from `kernel.tar.Z`. This creates files in the present working directory.
2. (Only when you wish to use host memory for expanded storage.) Apply the patch for reserved memory; update `/etc/lilo.conf`. (Refer to "Expanded Storage".)
3. Create `kernsrc.makefile` to define the macro L pointing to the kernel source tree. In modern kernels this can be found by looking in

   ```
   ls -l /lib/modules/$(uname -r)/build
   /lib/modules/2.4.22-ac4/build -> /gnu/kernel/linux-2.4.22-ac4
   ```

   With this setup, create this line in the `kernsrc.makefile`.

   ```
   L=/lib/modules/2.4.22-ac4/build/include
   ```

   Makefile macros do not support returning the output from a command; resist the temptation to simplify.
4. Issue `make -f p390.makefile cleanmod` to compile the kernel-dependent part of the device driver and create the object file. You should see no errors and no warnings with gcc version egcs-2.91.66; if you see any, something is incompatible in your setup. Warnings have been reported with more recent compilers/assemblers.
5. Insert the `loadp390` shell script in your inittab (or whatever your distribution uses).

   In a SuSE distribution, you would copy the file to `/etc/init.d` and you would create symbolic links from `/etc/init.d/rc2.d`. This example shows the SuSE 6.3 convention of using `/sbin/init.d`:

   ```
   #/sbin/init.d/rc2.d ls -l *p390
   lrwxrwxrwx   1 root     root            7 Oct 30  2000 K29p390 -> ../p390
   lrwxrwxrwx   1 root     root            7 Oct 30  2000 S29p390 -> ../p390
   ```

   SuSE 7.1 moved these files to `/etc`. I also had to put the links into `rc.3` (SuSE 7.2).

   ```
   #/etc/init.d/rc3.d ls -l *p390
   lrwxrwxrwx    1 root     root            7 Sep 23 17:49 K29p390 -> ../p390
   lrwxrwxrwx    1 root     root            7 Sep 23 17:49 S29p390 -> ../p390
   #/etc/init.d/rc3.d cd ..
   #/etc/init.d ls -l p390
   -rwxr-xr-x    1 root     root          843 Aug 28 18:34 p390
   #/etc/init.d
   ```

Expanded Storage

The distribution includes a patch that will apply to Linux 2.4.20 2.4.22 kernels to implement the boot parameter `slice=`. You should apply this patch before *e.g.*, the Alan Cox patch.

`manops` will configure host expanded storage automatically when memory has been reserved in this way; it issues messages to inform you of its actions.

You can reserve any number of megabytes, but the P390 supports only up to half a gigabyte of expanded storage.

You can enter the reserve parameter in the `lilo.conf` configuration file; here is a snippet from one of mine in the early days:

```
image = /boot/bzImage.2.4.4.slice
  root = /dev/sdd5
  append = "slice=256M"
  label = 2.4.4.slice
```

Refer to the beginning of `/var/log/boot.msg` to verify that the system has booted correctly:

```
<6>user-defined physical RAM map:
<4> user: 0000000000000000 - 000000000009fc00 (usable)
<4> user: 000000000009fc00 - 00000000000a0000 (reserved)
<4> user: 00000000000e0000 - 0000000000100000 (reserved)
<4> user: 0000000000100000 - 0000000030000000 (usable)
<4> user: 00000000fee00000 - 00000000fee01000 (reserved)
<4> user: 00000000fffc0000 - 0000000100000000 (reserved)
<4> user: 0000000030000000 - 0000000040000000 type 5
```

The last line is the result of slicing off a quarter of a gigabyte from the RAM available to Linux.

The sliced memory is taken from the end of the detected (or specified) physical memory (high addresses); it is not known to Linux, thus it does not enter into any of its calculations (it is as if you had specified `mem=` to set a particular memory size). In particular, with a standard kernel, *i.e.*, one that supports up to 1G memory, the reserved memory does not count towards the 1G limit (which is more like 900M these days).

If you have more than one P390 card in your machine, you can set up several slices. When the device driver initialises it allocates the largest slice available; as a result, you must name the devices in the same order as the PCI bus is scanned, which seems to be either left-to-right or the opposite, depending on the motherboard. You will have to bring up both systems (all?!) and query expanded storage to see whether you guessed right. If not, you have the option of swapping the cards. Refer to Appendix A, “Using Multiple P390 Cards in one System” on page 54.

My current kernel parameters are:

```
image = /boot/bzImage.2.4.22.slicea
 root = /dev/hda5
 append = "slice=256M slice=64M"
 label = 2.4.22.slicea
```

And the messages a boot are:

```
<6>Slice 10000000 at 2ff30000 type 0.
<6>Slice 4000000 at 2bf30000 type 0.
<6>user-defined physical RAM map:
<4> user: 0000000000000000 - 000000000009fc00 (usable)
<4> user: 000000000009fc00 - 00000000000a0000 (reserved)
<4> user: 00000000000e6000 - 0000000000100000 (reserved)
<4> user: 0000000000100000 - 000000002bf30000 (usable)
<4> user: 000000003ff30000 - 000000003ff40000 (ACPI data)
<4> user: 000000003ff40000 - 000000003fff0000 (ACPI NVS)
<4> user: 000000003fff0000 - 0000000040000000 (reserved)
<4> user: 00000000fecf0000 - 00000000fecf1000 (reserved)
<4> user: 00000000fed20000 - 00000000feda0000 (reserved)
<4> user: 000000002ff30000 - 000000003ff30000 type 5
<4> user: 000000002bf30000 - 000000002ff30000 type 5
```

Installation Verification

IPL a stand-alone card deck before going on to IPL an operating system. IPL DDRXA is supplied with the distribution. A device map needs to include a card reader and a console. You might also add a disk drive for it to test.

These are the steps:

1. Create device map.
2. Load kernel module or run `rescanp390` to define the devices.
3. Start `manops`.
4. Connect a 3270 emulator to the console port.
5. IPL the card reader.

Overview of Operating Procedures

Using the P390 involves several tasks that must be performed in this order:

1. Use the `loadp390` script to insert the device driver module into the kernel and define special subchannel files for the subchannels. This is normally done in one of Linux' initialisation scripts.
2. Run the `manops` command to initialise the card and load its operational microcode. The `manops` command must be active before device managers can be started. This is normally done in one of Linux' initialisation scripts.
3. Start device managers for the devices in the configuration. These device managers are usually started by `manops` when it reads the device map, but you can run device managers in separate sessions, as long as `manops` is running. One application of this is to attach a debugger to the device manager. `manops` terminates device managers it has started when it terminates; it is your responsibility to ensure that these other device managers are terminated in a timely manner before you terminate `manops`.
4. Use utilities to manage disk image files.
5. Perform recovery procedures when things go wrong.

Adding Subchannel Files Dynamically

You can add subchannel files dynamically either with the `rescanp390` command (when you have updated the device map) or by the `addp390dev` command, which does not refer to the device map. You cannot add the `manops` device dynamically.

Use the `addp390dev` command to create an additional subchannel file dynamically. The `loadp390` command must have been run previously to create the device major number and at least one subchannel file. Adding subchannel files can be done by `root` only.

```
addp390dev 371
addp390dev 371 p391
```

The first word specifies the device number; leading zeros are prefixed to bring its length to four characters. The second word specifies the device major name; `p390` is the default.

The file is created as `/dev/<major name>/<number>` with a subchannel number one larger than the largest number present. The major number, owner, and group are set corresponding to one of the existing devices.

```
#/root addp390dev 371 p391
#/root ls -l /dev/p391/371
ls: /dev/p391/371: No such file or directory
#/root ls -l /dev/p391/0371
crw-rw----    1 p391     p391     253,  22 May  5 10:16 /dev/p391/0371
```

Note that the device number is padded out to four digits.

Once the device is defined, you can add it to the `manops` configuration. This enables the subchannel. Once the subchannel is enabled, the device manager can be started. The device manager can be run stand-alone or under `manops`. You may also need to create a device media file.

This is a somewhat edited transcript:

```
p391@Linux:  > mkdir /var/p391/fba
p391@Linux:  > touch /var/p391/fba/370.media
p391@Linux:  > add device 370 3370 mgr3370 371 /var/p391/fba/371.media
p391@Linux:  > query dev 370
manopcmds152i Device type 3370 number 0370 subchannel 16 is idle.
manopcmds186i    Command: mgr3370 370 /var/p391/fba/371.media.
startmgr dev 370
Starting mgr3370 on subchannel 16 for device number 0370 with arguments:
   0370 371 /var/p391/fba/371.media
ibmsco012e 1 excessive argument(s) beginning /var/p391/fba/370.media.
Manager mgr3370 for device 370 ended exit status 126 20 active.
p391@Linux:  > mod device 370  mgr3370  /var/p391/fba/371.media
p391@Linux:  > startmgr dev 370
p391@Linux:  > q dev 370
manopcmds152i Device type 3370 number 0370 subchannel 16 is started.
manopcmds186i    Command: mgr3370 /var/p391/fba/370.media.
```

A channel report word is generated when a device manager is started for a device that was not processed from the device map. zVM acknowledges this on the VM console:

```
14:40:23 HCPRFC2264I Device 0370 is available and online.
```

Starting and Stopping the Kernel Module

The kernel module is `p390.o`. It is installed by the script `loadp390`.

The script has several synonyms that are implemented as hard links:

- `unloadp390` unloads the device driver and removes the subchannel files.
- `reloadp390` unloads the device driver and loads it again.
- `rescanp390` re-scans the device map and defines the devices now in the map. This will bring new entries to the device map online and take removed entries offline.

The device driver and the script are in p390's `bin` directory.

When the module is inserted, you can specify any of four arguments to the `insmod` command as arguments to `loadp390`; you can also specify these in the environment variable P390_MODULE_PARAMETERS. Note that the module parameters are not specified with the usual hyphen to indicate a flag.

Name	Type	Description
debug	int	Debug level. When this is increased over the default 0, the device driver module writes progress messages to the system log.
net	string	The name to use for one to four network interfaces to be defined. These network interfaces may be used by VM's CTC driver or attached to virtual P390/Linuxs. Specify the interfaces separated by commas, for example, `net=vm,lnx`. The default is the one interface `vm`. The actual interface used is constructed by appending the index of the p390 card as they are discovered on the PCI bus. Thus, for the first (or only) card, the default network interface is `vm0`. Note that all interfaces must also be defined in the device map so that they can be configured.
trace	int	The number of pages to allocate to the trace table. The default is 8 pages, which gives 1024 entries. Do not go overboard on this number as the amount of kernel space is limited. You have high-granularity control over what is being traced. Eight pages should be more than ample.
trace_mask	int	The default trace mask, which controls the level of tracing in the internal trace table. The default mask (`0x10000`) traces error conditions only. See "debug—Set Debug Level" on page 25 for setting the trace level from MANOPS.

Subchannel Files

In addition to the `manops` device, there is a special character subchannel file for each subchannel defined. The files are in the directory `/dev/p390`; the file name is the device number; the device minor number is the subchannel (0 through 254).

```
crw-rw----   1 p390     p390     254,   0 Aug 25 17:10 000c
crw-rw----   1 p390     p390     254,   1 Aug 25 17:10 000d
crw-rw----   1 p390     p390     254,   2 Aug 25 17:10 000e
crw-rw----   1 p390     p390     254,   3 Aug 25 17:10 03c0
crw-------   1 p390     p390     254, 255 Aug 25 17:10 manops
```

The subchannel device files can be opened by anyone in the group `p390`; they are at the level of the channel-to-control unit interface of olden days (the OEM interface). In particular, they allow no direct access to any facilities on the card. Programming errors will not cause harm outside the particular subchannel that is opened.

The `manops` device can be opened by the p390 user only because one can reach facilities directly on the card through this special device. This device is opened by the `manops` command.

If the device map is changed after the device driver module is loaded, the additional device names will not have subchannel files defined for them. Reload the device driver module or use `rescanp390` to cause new devices to be defined.

Service Processor

The service processor is implemented by the command `manops`. `manops` is a command interpreter that emulates the front panel of the processor.

`manops` initialises itself and the P390 through these steps:

- Load the functional microcode, if it is not already loaded.
- Read the device map up to the START statement and initialise the subchannels based on DEVICE statements.
- Start the device managers that run under the MANOPS process.
- Perform the device map statements after the START statement.
- Enable the “awscmd” REXX subcommand interface.
- Wait for operator commands on standard input and MANOPS alerts from the P390.

Unattended Operation

Specify `--input` referring to a named pipe and redirect standard output and standard error to run `manops` a server.

In this mode, `manops` reopens the input file whenever it sees end-of-file. This allows individual commands to be issued with *e.g.,* `echo`.

The output can be followed with the `tail -f` command.

`manops` Command Switches

You can specify these flags on the `manops` command:

Flg	Long name	Description
-d	--debug	Enable debugging.
-i	--input	File to be read for `manops` commands. The file may be a named pipe or a regular file. End-of-file is ignored on a named pipe. This allows commands to be sent with, *e.g.,* `echo`.
	--major	Specify major device name. The default is `p390`. Thus, the default `manops` subchannel file is `/dev/p390/manops`.
-m	--devmap	Specify the device map file to use. The default is `<major>.devmap` in the current working directory. If `-major` is omitted, the default device map is `p390.devmap`.
	--notimestamp	Messages from `manops` are not timestamped. This has no effect on device managers and started tasks.
-v	--verbose	More verbosity in messages.

Device Map

The device map file is read by `manops` when it starts. The `manops` ignores blank lines and lines beginning with an asterisk (`*`).

The device map file can be one or two sections that are delimited by the START statement. The first section is processed to define the devices and other system parameters; the second section is processed after the device managers have been started.

The first section of the file may contain these statements:

- A timezone statement that defines the timezone of the system.
- A host statement to define the host name to tn3270 servers. This should be the same as the system name in the system configuration file and the SYSTEM NETID file, not to mention the TCPIP configuration files.
- env statements to set environment variables; the variables are set as the statements are read.
- system statements which contain Linux commands; they are issued as the statements are read.
- Device statements that define the I/O subsystem.

The second section, if present, may contain these commands:

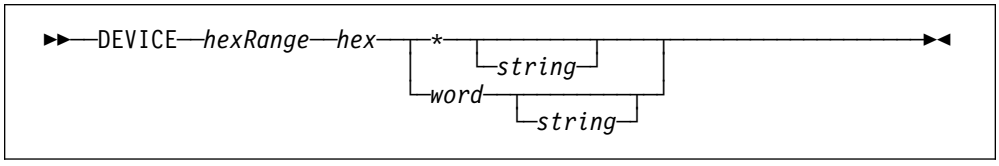
- manops statements to issue `manops` commands (see “Service Processor Commands” on page 24).
- env statements to set environment variables.
- system statements to issue Linux commands.

When the second part of the file is present, it should contain a system command to sleep a bit so that the device managers may initialise. The sleep time should be selected depending on the CPU speed, nature and number of device managers. You may also consider the `nice` command to lower `manops`' priority below that of the device managers.

In addition to the statements processed by `manops`, the device map also contains interface statements for network interfaces that should be started when the device driver module is inserted into the kernel.

device Statement

The `device` statement defines a subchannel.



DEVICE	Keyword; specify as written.
hexRange	Device number in hexadecimal (up to four digits). or a range of devices (3270).

hex	Specify the device type for documentation. The value is ignored.
*	Indicates that the device manager will run in a separate process. Further comments may be specified.
word	The executable for a device manager that is to run in the service processor address space.
string	Command-line parameters for the device manager.

Device Manager Parameters

The parameters for the device manager are specified in the standard Linux format of flags and positional arguments. Of these, some are standard for the device manager infrastructure and others are specific to the device driver at hand. For the latter, refer to “Device Managers” on page 34.

For some flags, a default may be supplied as an environment variable.

Standard Flags

Flg	Long name	Environment variable	Description
	--ccwin		Specify the name of a file that contains CCWs, data, and operator commands to drive the device manager offline. `--ccwin` is intended for testing device managers.
	--ccwout		Specify the name of a file to receive CCWs, data, and operator commands that the device manager has seen. This provides a detailed trace of the device manager’s interaction with the subchannel file. With minor editing, such a file can be used to regression test with the `--ccwin` flag.
-d	--debug		Enable debug mode. In the debug mode, the infrastructure and the individual device managers display additional information about the progress of operations. Specifying this flag multiple times will increment the debug level. `--debug` implies `--verbose`.
	--devicepath	P390_DEVICEPATH	The path to the directory that contains the special subchannel files representing the subchannels. The default is `/dev/p390/`.
	--directory	P390_DIRECTORY	Specify the working directory for the device manager.
	--logfile		Specify the file where the device manager logs all data moved to and from the subchannel file. This flag is used for debugging only.
	--notimestamp		Messages are not timestamped.

Flg	Long name	Environment variable	Description
	--sync		Write the trace file synchronously. This flag allows you to see the last device manager action before a bus hang. This flag can slow down considerably the creation of large amounts of debug trace,
-t	--tracefile	P390_TRACEFILE	Specify the trace file name. The full trace file is constructed from the trace path and the file name. See below for a discussion of the trace file. The default trace file is the device number (`%n`).
-t	--tracepath	P390_TRACEPATH	Path to the directory where trace files are stored. The default is `/var/p390/trace`.
-v	--verbose		Issue more progress messages than otherwise.

The Trace File

Specify a hyphen (`-`) for the trace file to remain the default, which is standard error.

When writing the trace file to disk, you can specify substitutions in a way similar to the `printf` subroutine:

%n The device number, e.g., 000C

%s The subchannel number in decimal, e.g., 0.

%p The process number in decimal.

%d The date as an eight-digit standard number, e.g., 20000319.

%t The time as a six-digit number, e.g., 101106.

Specify two percent signs should you desire one such in a file name.

The default trace file is `/var/p390/trace/%n`.

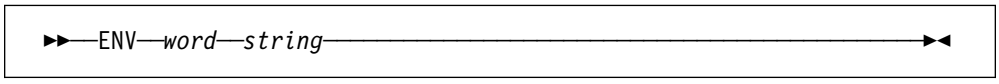
Standard Positional Arguments

`manops` prefixes the device number (range) to the command it issues to start a device manager. This represents the special file that is connected to the System/390 subchannel.

Device managers that use Linux files usually specify the file name as the second positional argument. Further arguments and flags depend on the particular device manager; see “Device Managers” on page 34.

env Statement

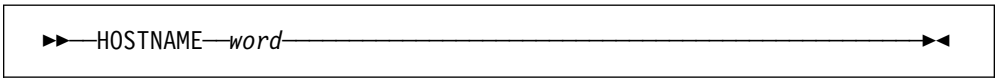
Use the env statement to set an environment variable in the environment that will be available to device managers started by `manops` and to started tasks.



The word and string are joined with an equal sign and added to the environment.

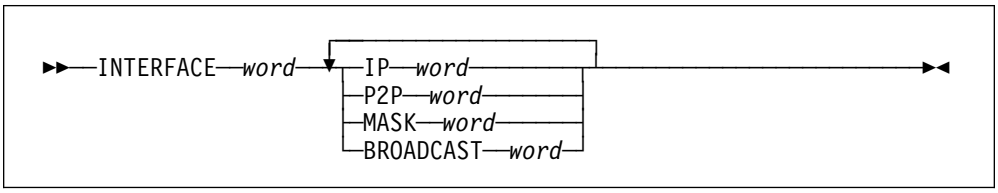
hostname Statement

Use the hostname statement to specify the host name that will appear when clients connect to the TN3270 server for local devices.



interface Statement

Use the interface statement ot configure the network interface to be used in attaching the VM TCP/IP stack to your network.



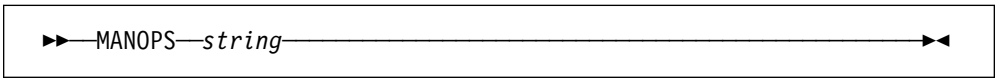
This statement is processed when the device driver is loaded into the kernel; any errors are reported at that time as well.

You must specify IP and P2P. IP specifies the IP address on the Linux system; P2P specifies the VM system's IP address (strictly, the IP address associated with the channel-to-channel device configured on the 390 side).

If you omit the MASK and/or BROADCAST operands, those will be computed based on the network class.

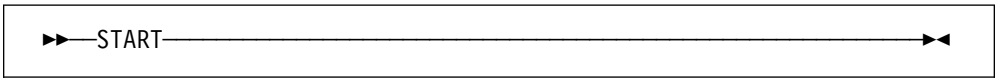
manops Statement

Use the manops statement to issue a manops command as if you had entered it on the standard input. The command is issued immediately.



start Statement

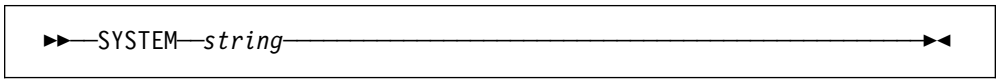
Use the start statement when you have finished defining the system configuration and wish the device managers to be started. The remainder of the device map (which may now be considered a profile) is read after the device managers have started, but since the device managers run in separate processes, it is not given that they will have initialised immediately.



No operands are allowed.

system Statement

The command is processed by Linux through the `system()` library function as it is read from the input file. When `-verbose` is specified, the command is logged to the standard output file.



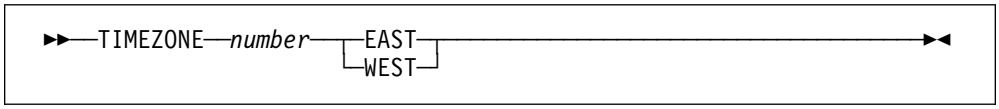
Example

To delay execution of the `manops` process for a second, for example to allow device managers time to start:

```
system sleep 1
```

timezone Statement

While any number of timezone statements may be specified, only the last one will have any effect.



The number specifies the number of minutes of difference from the local time to GMT (by some called UTC).

Example

For Copenhagen during the winter, this would be appropriate:

```
timezone 60 east
```

Poughkeepsie NY might want this during the winter:

```
timezone 300 west
```

Example Device Map

```
timezone 60 east
hostname CPHART

* "device" address type ...

device  00c     3505    mgr3505         IPL.DDRXA
device  00e     1403    mgr1403         -replace /var/p390/printers/00e.out
device  01c     3505    mgr3505         IPL.ICKDSF

system rm /var/p390/trace/*

* SFS pack.

device  125     3390    mgr3390         \
        -vol 240w02 -pincount 13 -d

* zVM

device  310     3390    mgr3390 -vol 310res -d -pincount 5

device  311     3390    mgr3390 -vol 310w01 -d -pincount 5

system awscke -create -out /var/p390/drives/310p01.cke -cylcount 1000 \
                        -page -label 310P01

device  312     3390    mgr3390 -vol 310p01 -d

device  313     3390    mgr3390 -vol 310s01 -d -pincount 5

* Private data

device  300     3390    mgr3390 -vol john -d -pincount 3
device  301     3390    mgr3390 -vol zvm44 -d -pincount 1

* Linux drives
device  200     3390    mgr3390         \
        -vol lnxswp   -d

device  201     3390    mgr3390         \
        -vol lnx001   -d

* Tape drives
device 180      3423    mgr3423         \
        -dir /var/p390/tapes/ tape.0180 -format jph -d
device 181      3423    mgr3423         \
        -dir /var/p390/tapes/ tape.0181 -format jph -d

* VM console
device  01f     3270    mgr3270         \
        -p 3270 -b 0                    \
        -dir /var/p390/work/001f        \
        -ipldev 310

* Linux console.  Avoid 9 as that is the default 3270 console for VM
device  008     3215    mgr3215         -a -d

* ctc devices for network access
device  3e0     3088    mgr3088c        \
```

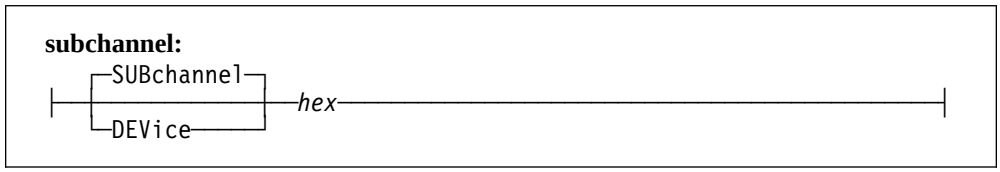
```
        -d -netif vm -writeside

device  3e1     3088    mgr3088c        \
        -d -netif vm

device  3e2     3088    mgr3088c        \
        -sync -d -netif lnx -writeside

device  3e3     3088    mgr3088c        \
        -sync -d -netif lnx

device  3c0.8   3270    mgr3270 -v \
        -p 3274 -holdtime 3600 -maxhold 2

device  400     3088    mgr3088 \
        -host linux -port 20000 -listen 20001 -d

device  401     3088    mgr3088 \
        -host linux -port 20001 -listen 20000 -d

* link to P391.
device  500     3088    mgr3088 \
        -host linux -port 20500 -listen 21500

device  501     3088    mgr3088 \
        -host linux -port 20501 -listen 21501 -d

device  502     3088    mgr3088 \
        -host linux -port 20502 -listen 21502 -d

interface vm0                   \
        ip 10.0.1.2             \
        p2p 10.0.1.1            \
        mask 255.255.255.0      \
        broadcast 10.0.1.255

interface lnx0                  \
        ip 10.0.2.2             \
        p2p 10.0.2.1            \
        mask 255.255.255.0      \
        broadcast 10.0.2.255

start

system sleep 2
manops debug p390 reset
manops system clear
* manops debug p390 on ffff
* manops debug dev 180 on ffff
* manops debug dev 181 on ffff
* manops debug dev 400 on ffff
* manops debug dev 401 on ffff
manops debug dev 500 on ffff
* manops debug dev 3e0 on ffff
* manops debug dev 3e1 on ffff
* manops debug dev 310 on ffff
* manops debug dev 3c0 on ffff
manops starttask sad mgrsad -port 3271
```

Service Processor Commands

Once manops has initialised itself, it waits for commands on its standard input. The valid commands are described in the following sections.

General Constructs

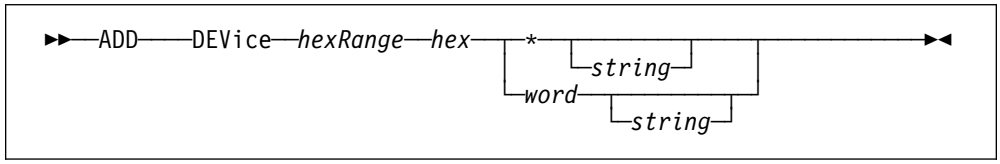
The syntax fragments below are used in various places.



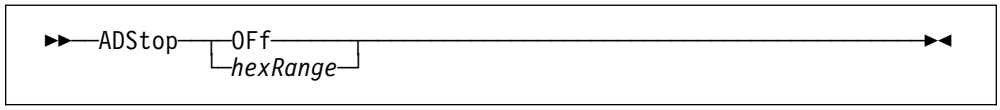
A subchannel is referred to by a hexadecimal value. The hexadecimal value may be a device number or a subchannel number; when the keyword is omitted, the number is taken to be the subchannel number.

add—Add a Device Dynamically

Add a device to the configuration while manops runs. The device manager is started if a command is specified. If a device range is specified, the devices must have consecutive subchannel numbers. Refer to “device Statement” on page 17 for further information.

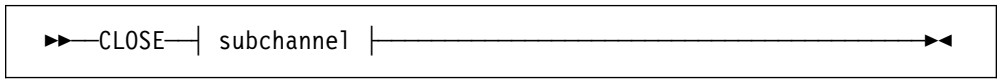


adstop—Address Compare



close—Force Close of a Device Manager

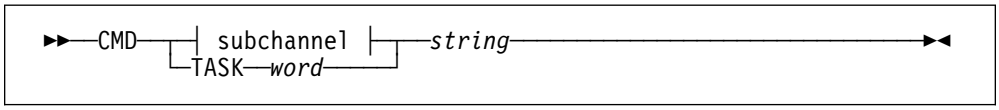
The CLOSE command is used to recover a device after the device manager process has been terminated in the device driver. It forces a subchannel to the idle state and decrements the use count of the device driver module.



Warning: The CLOSE command is intended to alleviate the need for a reboot to recover from an exception in kernel mode. Casual use may lead to system instability, loss or duplication of data and all kinds of evils.

cmd—Issue Command to Device Manager

The `cmd` command is sent to a device manager task for processing. This can be done only for device managers that are started by `manops`.



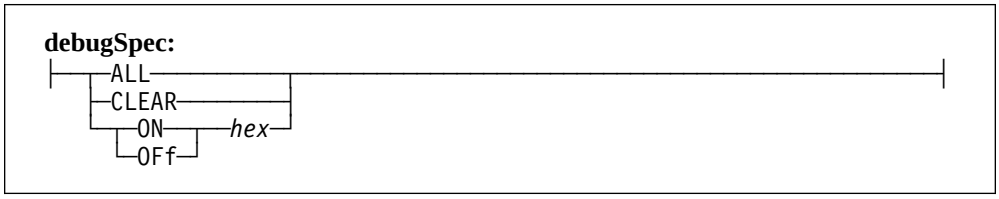
word — The name of a started task.

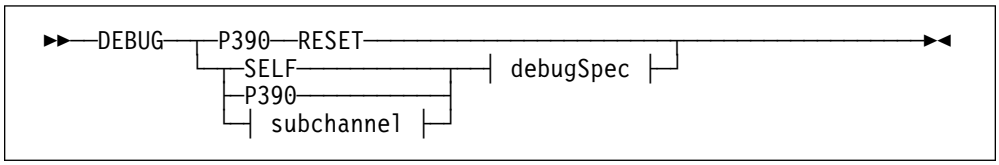
string — The command to be sent to the device manager or task. The string extends to the end of the line.

debug—Set Debug Level

The `debug` command enables or disables debugging at various places:

- In `manops` itself.
- In the P390 device driver in general.
- In device drivers for individual subchannels.

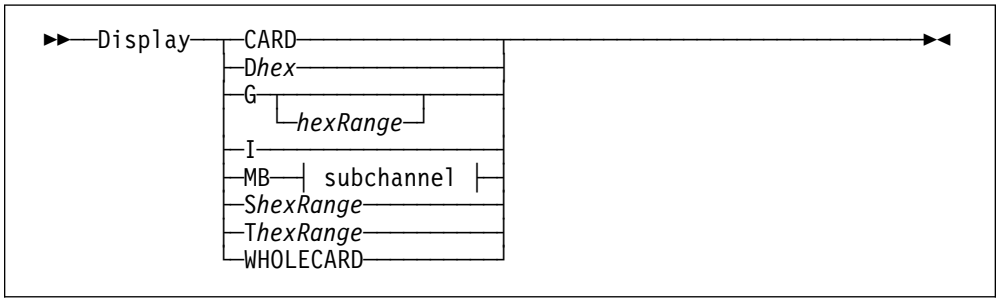




`debug p390 reset` clears the internal trace table.

d—Display Processor Storage and Subchannels

The command format follows CP conventions, but is most definitely a subset. Unlike CP, it is not enforced that the formatting is a single letter, but if the parser finds something it does not understand it stops and you will see the display for number 0.

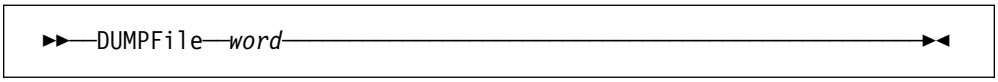


The modifiers are:

CARD	Dump the first part of the contents of the master control block in the device driver.
d	Device block. The control block representing the subchannel in the device driver. This function is intended for debugging the device driver and other diagnostic purposes.
g	General registers.
i	The four Interrupt Control Blocks.
MB	Measurement block for the specified subchannel.
s	Subchannel.
t	Processor storage. The display shows sixteen bytes per line with both ASCII and EBCDIC interpretations.
WHOLECARD	Dump the complete contents of the master control block in the device driver.

dumpfile—Dump Contents of Main Storage to File

The `dumpfile` subcommand writes the entire contents of storage to the specified file. The core image may be restored with “loadfile—Load Binary Data Into Storage” on page 28.

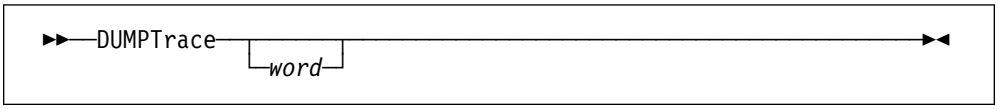


Notes:

Usage notes:

1. The system should be in the stopped state to ensure that a consistent view of storage is obtained.
2. The storage protect keys are not saved.

dumptrace—Dump Device Driver Trace Table to File

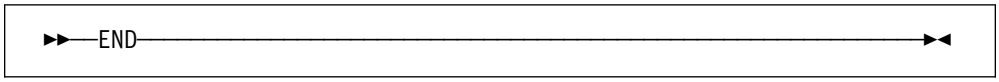


The word is the file to write. If it is omitted, the dump is written to `/var/p390/trace/dump`.

The trace table entry is thirty-two bytes. The amount of storage reserved for the trace table is specified when the kernel module is installed.

Use the utility `fmttrc` to format the file.

end—Terminate Service Processor

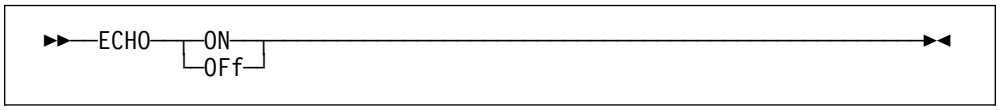


A stop command is sent to all active device managers that run under control of `manops`. `manops` then waits for up to ten seconds for them to stop.

`manops` then performs the quit function; see “quit—Terminate MANOPS” on page 30.

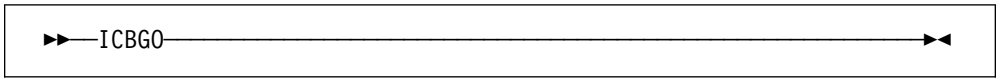
echo—Control Echoing of Commands to Standard Output

The `echo` subcommand sets echoing mode on or off. Echoing is useful when the standard output is redirected to a file.



icbgo—Release Interrupt Control Block

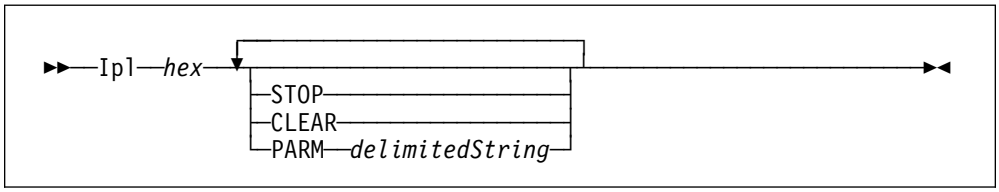
The `icbgo` subcommand releases the Host Alert Interrupt Control Block. This may be used to recover from a programming error in manops that causes a hang by refusing to perform the requested action or to recover when an unsupported alert is received.



Warning: Unscrupulous use of ICBGO may interfere with manops processing and cause all kinds of errors.

ipl—Initial Program Load

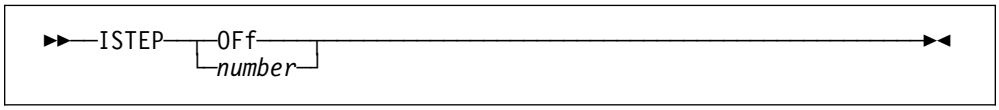
The IPL command loads the operating system into the machine.



You may also IPL from the 3270 device that is connected to the device manager for the console if it is configured accordingly.

istep—Instruction Step

The ISTEP command activates or terminates instruction step mode.

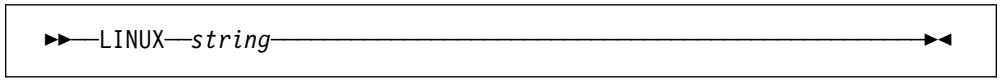


The number specifies the number of instructions to execute before returning to stopped state.

Use the START command to resume instruction execution after `manops` reports instruction step complete.

linux—Issue System Command

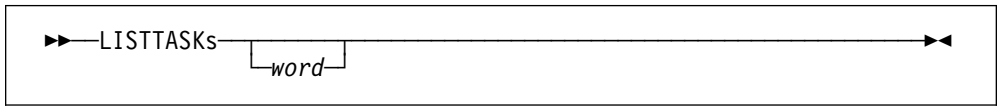
The command is issued via the `system()` library function.



Note: As the `system()` function interferes with `manops`' harvesting of SIGCHLD signals, the usage of the `system manops` subcommand is discouraged.

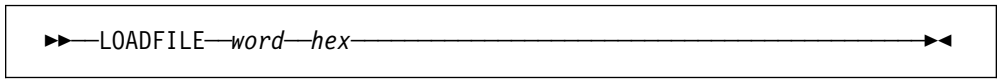
listtask—List Active Started Tasks

If a task name is specified, information for the particular started task is displayed; otherwise information for all started tasks is displayed.



loadfile—Load Binary Data Into Storage

The specified file is loaded into storage at the specified address as if its contents had been entered by store commands.



Example:

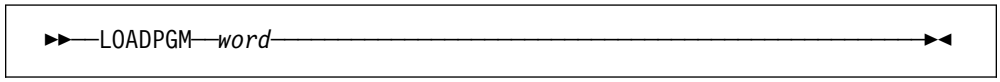
```
system clear
loadfile /cdrom/suse/images/tapeipl.ikr 0
loadfile /cdrom/suse/images/initrd 800000
loadfile /cdrom/suse/images/parmfile 10480
system restart
```

Nota bene: Unlike the suse.ins file, do not specify 0x with the address.

loadpgm—Load Program Into Storage

The specified IPL deck is loaded into storage as if you had entered it with store commands. The first eight bytes of the first record are loaded into the PSW.

LOADPGM was written to debug the core of the device driver; since card readers are now supported, stand-alone programs can be IPLed directly.

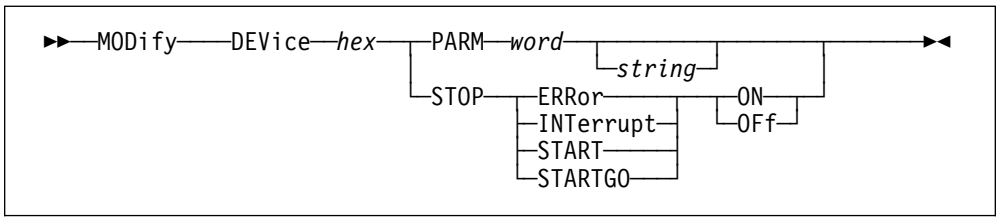


word — Specify the file name of the IPL deck file. The file type `ipldeck` is appended by Manops.

The IPLdeck is in a special format; it is generated by a CMS utility from an object module.

modify—Modify Device

Store a new command string for a device. For a group of devices, use the first device number.



PARM — Specify program name and parameter string for device manager. The options string should omit the device number; this is supplied automatically.

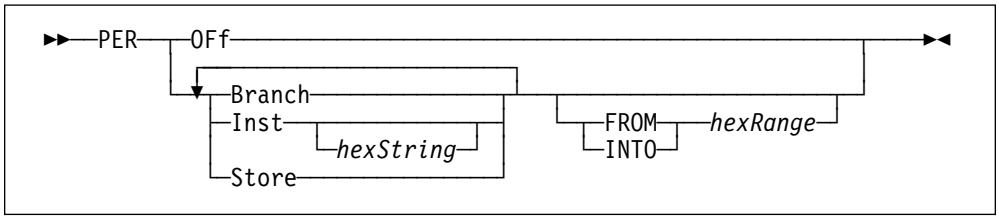
```
mod device 371  mgr3370  /var/p391/fba/371.media
```

STOP — Change P390 subchannel stops. When enabled, the CPU enters the stopped state after the specified event occurs. Use the START command to resume processing.

- ERRor — Stop when an error conditions is reflected on the subchannel; any of the device and channel status flag bits X'037F' are on.
- INTerrupt — The P390 enters the stopped state after an interrupt has been presented.
- START — The P390 enters the stopped state before a SSCH instruction is executed. That is, the channel program is not started.
- STARTGO — The P390 enters the stopped state after a SSCH instruction has been executed successfully. As this lets the channel program run, the practical utility may be limited.

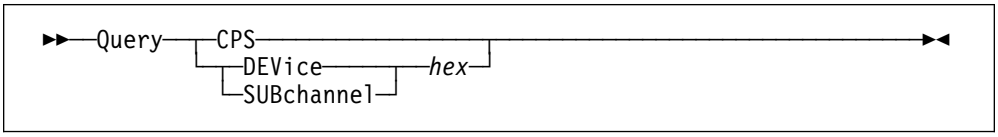
per—Program Event Recording

Enable trace of program. Use PER to trace real program execution. This command is designed to debug an operating system; use the native TRACE or debuggers once the operating system is operational.



Unlike the VM PER command, the manops per command specifies the all the options that apply; manops does not merge the options into an existing trace set.

query—Obtain Information

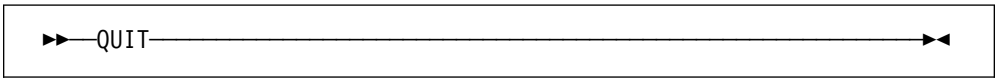


CPS — Display the channel path status for the eight CPIDs supported by the P390. The numbers displayed are the number of active channel programs on each CPHID. As P390/Linux uses only CPHID 0, all others should be zero.

DEVice — Display information about the device manager active for the specified device number.

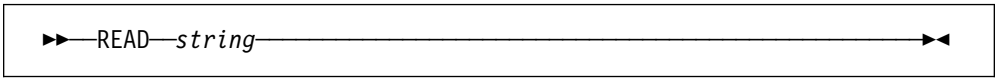
SUBchannel — Display information about the device manager active for the specified subchannel.

quit—Terminate MANOPS



manops terminates without sending a stop command to the device managers; they are send the SIGTERM signal to terminate.

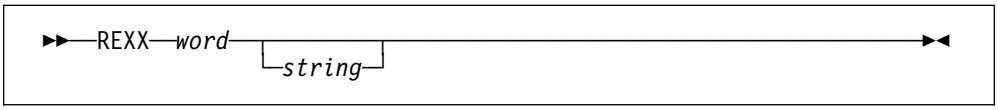
read—Read MANOPS Commands from a File



The input is diverted to the specified file. When the commands in the file have been read, processing reverts to the interrupted file or standard input.

rexx—Run a REXX Command Script

The REXX command runs a REXX program with manops as the default command environment.



The word is the path to the program to run. The word is the full file name; nothing is added.

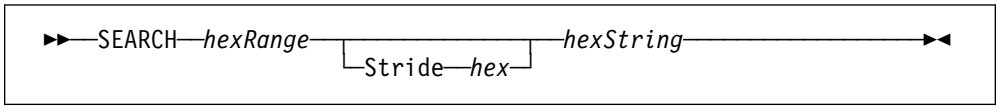
The string is passed as the one argument string to the REXX program.

The REXX program runs within the manops process and may change the manops environment permanently (see “Using REXX” on page 33).

Note: REXX issues normal commands (address command) using `system()` library function. As the `system()` function interferes with `manops`' harvesting of SIGCHLD signals, the usage of Linux commands in a `manops` REXX script is discouraged.

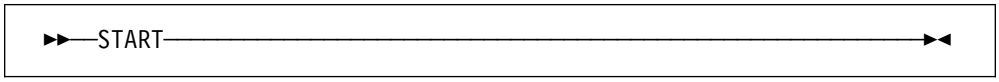
search—Search Main Storage

Search core storage for a string.



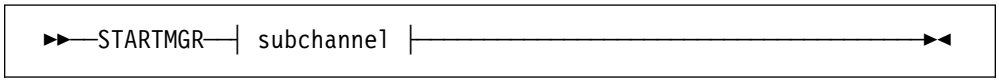
xrange	Specify the range to search. The addresses are in absolute storage.
hex	Specify the increment between compares. The default is one.
hexString	Specify hexadecimal data to look for. You must specify an even number of hexadecimal characters. The maximum search string is eight bytes (sixteen hexadecimal digits).

start—Resume Instruction Execution



startmgr—Start a Device Manager

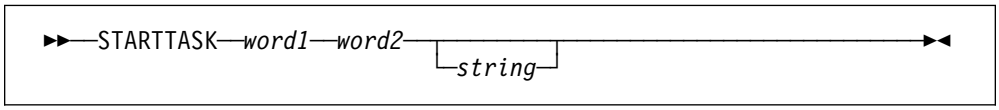
The device manager defined for a specified subchannel is started as a child of the Manops process



subchannel	Specify the subchannel number in hexadecimal or the keyword DEVice followed by the device number in hexadecimal.

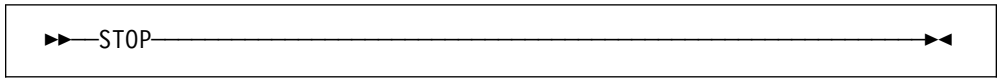
starttask—Start a General Task

The specified command is run as a child of the Manops process.



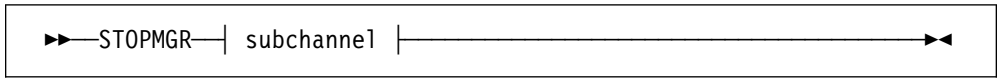
word1	The first word specifies the name under which the task is known. The task name must be unique.
word2	The second word specifies the name of the program to run.
string	The arguments to the command, if any.

stop—Suspend Instruction Execution



stopmgr—Terminate Device Manager

The specified device manager task is signalled to terminate.



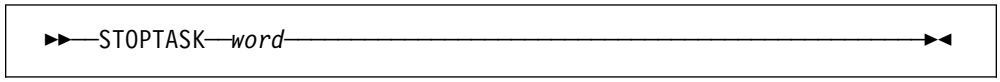
subchannel Specify the subchannel number in hexadecimal or the keyword DEVice followed by the device number in hexadecimal.

Notes:

1. Use `cmd device <number> stop` to terminate a device manager gracefully.

stoptask—Terminate General Task

The specified task is signalled to terminate.

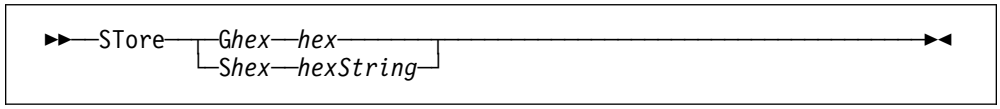


word The name under which the task was started.

Notes:

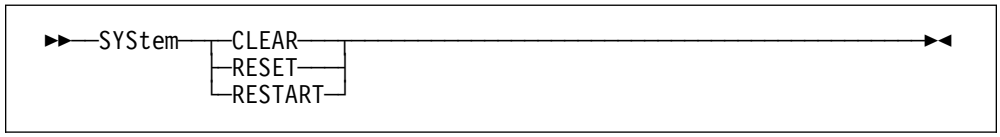
1. Use `cmd task <name> stop` to terminate a task gracefully.

store—Store into Registers or Processor Storage



system

The SYSTEM command performs resets, *etc.*, similar to the CP commands.



Using REXX

The operator can invoke a REXX script directly by the REXX command or indirectly from a script that is invoked through the LINUX command. In both cases can the REXX program address commands to the “awscmd” subcommand interface.

In the former case, the REXX program runs within the MANOPS process. Commands that modify MANOPS state (*e.g.*, DEBUG SELF) will have a lasting effect. In the latter case, the REXX program will run in a separate process and it will not be able to modify the state of the MANOPS process itself.

In both cases, the REXX program will be able to modify the state of the P390 and of device managers that were started from MANOPS through the `cmd` subcommand.

The underlying interface has no facility to return a return code; as a result, the return code is not set by the subcommand processor.

Nota bene: A REXX program that is not invoked in the MANOPS process should not issue STARTMGR commands. While the command will appear to work, as the information about the subprocess is lost and the effect will be as if the device manager were started in a separate session.

Example

This program loads a Linux kernel and parameters into storage and then starts the install process:

```
system clear
loadfile /cdrom/suse/images/tapeipl.ikr 0
loadfile /cdrom/suse/images/initrd 800000
loadfile /cdrom/suse/images/parmfile 10480
system restart
```

Sample

```
p/home/john/src/p390: manops
ucode.c: Loading microcode version V01.28 from AWS39X0.MCD.
ucode.c: read line 0, size 24
ucode.c: Loaded microcode.  Last word 12706.
devices.c: I/O subsystem ready.  0x3 highest subchannel.
manops.c: PSW:  mask 00 key 0 xmwp 0000 0000 24bit IAR 00000000.   PCS:
Processor state:  Stopped.
```

Device Managers

Device managers interface the subchannel to device media files.

The parameters (flags) when invoking a device manager are specified on a `device` statement in the device map.

Sample device maps are supplied with P390/Linux. Refer to `sp390.devmap` and `sp391.devmap` in the installation directory.

General

Device managers are started by `manops` when the program name is specified in a device statement or by the `manops` subcommand `startmgr`.

Arguments

The first positional argument to a device manager is the device number or device number range to be serviced by the device manager. This is specified in hexadecimal.

For device managers that use a single device media file, such as mgr3423, the second positional argument specifies the file to use or a pattern to generate the file name. Other device managers, notably `mgr3390` support a richer set of device media files.

Refer to “Standard Flags” on page 18 for command arguments that are common to all device managers.

Common commands

Device managers and started tasks respond to commands on their standard input. The device manager infrastructure deletes leading and trailing whitespace from the command string and routes it via a table in the individual device manager.

When the first non-whitespace on the line is one of the characters “,./?!”, the command verb is that single character; otherwise the command verb is the first whitespace-delimited word of the line.

Case is ignored in command verbs.

Device managers that run under control of manops receive their commands through a pipe; use the CMD manops command to route a command to such a device manager.

debug	Turn on debug mode in the device manager.
nodebug	Turn off debug mode in the device manager.
stop	Terminate gracefully.
superdebug	Turn on intensive debug mode in the device manager.
system	Execute the argument as a Linux command.

Zipped Files

The device manager infrastructure will unzip a zipped input file transparently. This is intended for the `awscke` utility (see “awscke—3390 Disk Utility” on page 44), but could also be used for reader files or tape files. It is, of course, a requirement that the file be read sequentially; this is enforced.

mgr1403—Line Printer

This device manager responds to 1403 command codes and generates an output file in ASCII that has ASA carriage control. Each line is ended by a newline character (`X'0A'`). Nonprintable characters are substituted with the tilde. The 390 code page is 1047 (US); the Linux codepage is 819 (Latin-1).

Flags

Specify the output file as the positional argument.

Flg	Long name	Description
-r	--replace	The printer file replaces the contents of any existing file. The default is to append to the file, if it already exists.

Commands

The 1403 responds to these commands:

file	Specify the file to receive the printed lines. The current output file is closed. If the -replace flag is in effect, the new file is truncated.
rewind	Discard the contents of the current printer file and continue printing to it.

mgr3088—Channel-to-channel Adapter

The channel-to-channel adapter connects subchannels on the same or different p390s.

Two instances of `mgr3088` are needed to implement the adapter. The two managers can communicate over:

- TCP/IP streams.

Flags

Flg	Long name	Description
	--host	Specify the host name of the system where the partner device manager is running. This selects TCP/IP as the communications vehicle. The `-listen` and `-port` flags are required.
	--listen	Specify the port that this instance of the device manager should listen on. This port number must be specified with `-port` on the other side of the adapter.
	--port	Specify the port that the device manager on the other side of the adapter is listening to. This port number must be specified with `-listen` on the other side of the adapter.

mgr3088c—Channel-to-channel Adapter for Network Interface

This device manager makes VM's (or Linux/390's) IP stack look like an ethernet card to the host Linux/i386.

Flags

Flg	Long name	Description
-n	--netif	Specify the symbolic network interface name without the trailing sequence number, for example, `-net vm`.
-w	--readside	The CTC is from which the P390 stack will read; thus, this is the read side as seen from VM's stack; or the read channel in Linux for System/390 parlance. If this flag is specified for the even device number, the VM stack should specify the third parameter of the LINK statement as 0. `--writeside` is an obsolete name for this flag.

mgr3215—3215 Printer/keyboard

While there are no IBM 390/ESA operating systems that support a 3215 and real 3215s no longer exist, this manager is implemented for Linux/390, which does not support a 3270 operator console.

Flags

Flg	Long name	Description
-a	--autocr	Append a carriage return to all output lines written by write-no-carrier-return when the line does not end in a line feed.

Commands

The 3215 responds to these commands:

.	(Period.) Provide a line of input. The 3215 end key is pressed at the end of the line. “i” is a synonym for this command.
r	Press the request key.

You cannot enter multiline input directly, but Linux/390 supports escape sequences using the caret as escape character; for example, `^n` represents a line feed.

mgr3270—3274-41D Control Unit

This manager is a tn3270 server. Connect to it with a TN3270 client that supports TN3270E.

Typically, you would have one 3270 device manager for the operator console and another one for a number of local 3270s for general use. In a hostile environment you will want to password protect the operator console, but not necessarily the general use 3270s.

The IPL function is available only when the device manager runs as a child of the MANOPS process.

Flags

Flg	Long name	Description
-b	--backlog	Specify the backlog of logon requests that TCP/IP should allow to queue up.
	--data	Trace data sent to and received from the TN3270 client.
	--holdtime	Specify the number of second before closing a socket that has been refused access because there are no free devices on the control unit. The default is 10.
	--ipl	Enable the IPL panel at connect.
	--ipldev	Specify the default IPL device number. This implies `-ipl`.
	--maxhold	Specify the maximum number of sessions that wait for a device at any one time. The default is 32, which is also the maximum value.
	--password	Specify the password that must be supplied for data to be transmitted to the host.
-p	--port	Specify port to listen on. Default is port 3270.
	--test	Test mode. The driver runs stand-alone (without a p390 device).
-v	--verbose	Additional information is written to the trace file.

When client session have been refused because there are no available devices, the connections that cannot be serviced are kept in a queue until the hold time expires or a device becomes available.

mgr3370—FBA Disk Drive

This manager implements a 3370-1 drive. It takes no additional flags and no additional commands.

To create a 3370 volume:

1. Use the `touch` Linux command to create an empty device media file.
2. Attach the empty device to a CMS user.
3. Use the CMS FORMAT command to format the device.

The volume now has a label; it can be attached to SYSTEM.

mgr3390—ECKD Disk Drive

This manager implements a 3390 drive. This drive has a capacity that is larger than 2G; as a result, two files are used by the original P370 implementation. You are urged to convert the files to the CKE_P370 format to conserve disk space and improve performance.

Note: `mgr3390` supports two formats of device media files:

CKD The R390 implementation. The file contains unblocked and uncompressed tracks. Once created, the file does not increase in size. Specify `--twofiles` when the disk pack is larger than 2G bytes (as is a model-3 pack).

Pinned files are not supported with this format. Specify neither `--pincount`, `--pinned`, nor `--volume`.

CKE The P390/Linux preferred format. The file contains squished and compressed tracks. A blank track (no R1, CP or CMS format) is represented by four bytes. The view of the device can be versioned through the use of pinned files. Disk storage requirements in general increase as a volume is filled with data.

The `awscke` utility creates blank volumes; it can also be used to convert from CKD to CKE format. See “awscke—3390 Disk Utility” on page 44.

General Use Flags

Flg	Long name	Description
	--datadirectory	Specify the directory in which the disk image files reside. This flag works in conjunction with `-volume`. If `-datadirectory` is not specified, the environment variable P390_DRIVES is inspected and used if present; otherwise, the default. `/var/p390/drives`, is used.
	--pincount	Specify the number of pinned files. The file format must be CKE. A new file is created when the number is larger the number of existing files. Do not specify a number larger than one plus the number of existing files. This flag has no abbreviation.
-r	--readonly	Reject write operations to the device. This forces a file mask that will prevent writes.
	--volume	Specify the volume part of the file name for the disk image files. The base file will be `<datadirectory><volume>.cke` and the pinned files will be `<datadirectory><volume>_%d.cke`. If `-pincount` is omitted, the number of pinned files will be determined by looking for them.

Esoteric Flags

Flg	Long name	Description
-c	--cache	The number of tracks in the control unit cache. The minimum number is one; the default is 128; the maximum is 32K, which would allocate slightly less than 2G bytes—not recommended.
-w	--writesync	Perform all writes to the device media file as synchronous writes. You need this flag if you use SFS or other highly reliable functions that depend on physical write having been performed at the end of a channel program. It is questionable whether Linux actually performs the `synch` function.

Debugging Flags

Several of the flags documented here are intended for debugging. It is recommended that you use pinned files to hold the modified tracks, as this may make smaller back-up files.

Flg	Long name	Description
	--tracecylinder	Specify a cylinder number. A detailed log of activity to the specified cylinder is written to the trace file.

Obsolete Flags

Flg	Long name	Description
-p	--pinned	Specify the full path and file name of the file to receive pinned tracks. Specify "%d" where you wish the pinfile's sequence number substituted. When -pinned is specified, the base device media file is opened read only, they cannot be modified. This flag has no abbreviation. **Note:** `--volume` is the preferred way to define pinned files.
-b	--twofiles	Two files are used to represent the drive. The device media file is in the original P390 format (AWSCKD). Specify %d in the file name to represent the file sequence number.

Example

```
device  124     3390    mgr3390         \
        -vol 240w01 -d                  \
        -pincount 14
```

Commands

The 3390 responds to these additional commands, all of which are for debugging use:

ckcache	Verify the cache index' integrity.
dumpstats	Display cache statistics.
dumptrack	Dumps the contents of a track to the trace file. Specify the track address in hexadecimal. You are likely to obtain the track address from the "crtk" or similar "usr" trace record, where it is hexadecimal.

Note: The 3390 device manager implements a write-through cache. Updated tracks are written to the pinned file or the original file no later than the completion of the channel program in which they were modified. Linux will delay the writing of data to the physical disk unless `--writesync` is specified. It is questionable whether Linux implements synchronous writes.

Pinned Files

On a real 3990, the term pinned refers to track data that are kept in non-volatile storage because the disk track has become unusable. In this implementation, pinned tracks are written to a different file than the one whence they were read, because the original file has been made read only by specifying `-pinned`.

While a pinned track contains as much data as the original track, the use of pinned tracks allows for easy restore of a previous disk and for trivial reversal to the original disk contents (delete the pinned files).

mgr3423—Tape Drive

This manager implements something close to a 3422. You must specify the format of a new tape file; for an existing tape, the format is determined from the file.

Currently the supported format is `jph`.

Flags

Flg	Long name	Description
	--format	Specify tape format. See below for acceptable values.
-r	--readonly	File protection ring is removed. Write-type commands will be rejected.

Supported Tape Formats

jph — Tape to be created is in a private format particular to the P390/Linux implementation.

Commands

The 3423 responds to these commands:

mount	Unload the current tape file and mount the specified file. A new tape is created if the file does not exist.
rewind	Rewind the tape to the beginning.

mgr3505—Card Reader

This manager implements a card reader that is able to IPL stand-alone programs such as IPL DDRXA, and also files generated in a special format on VM. The sample file is `ripstore.ipldeck`, which sets all words above the prefix area to contain their respective address.

Flg	Long name	Description
	--id	Specify the VM user ID to receive the card files read by the reader. The device manager generates an ID card internally.

The second positional argument is the name of the card deck. When the `-binary` flag is specified, it is in a rather special format, generated by a VM utility from an object deck.

Commands

The 3505 responds to these commands:

deck	Specify the a new input file. The file is rewound.
rewind	Continue reading from the beginning of the file. Remember to rewind the card reader if you wish to IPL the first deck again.

Started Tasks

A started task runs as a child of the `manops` process. Unlike a device manager, a started task cannot run in a separate session; it is started by the `manops` subcommand `starttask` (see “starttask—Start a General Task” on page 31).

Started tasks support the common device manager commands. See “Common commands” on page 34.

mgrsad—Server for System Activity Display

`mgrsad` is a TCP/IP server from which the System Activity Display client obtains the load statistics of the P390. The counters are:

- Supervisor state.
- Problem state.
- Wait state.
- Time waiting for `manops`.
- Emulate time; time spent in SIE.
- Supervisor state key 3; active wait.

When the P390 is operating, one of these counters is incremented every 3.3 millisecond.

Flags

Flg	Long name	Description
-d	-debug	Increment the debug level.
	-port	The TCP port on which `mgrsad` listens for its clients to connect. The port number must be larger than the highest privileged port number.

Writing a Client for mgrsad

When the client transmits a packet that contains three bigendian unsigned integers that contain:

1. 12, the total length of the packet.
2. 8, the total length of the record.
3. 1, for request SAD counters.

That is, the packet contains in hexadecimal:

```
0000000c 00000008 00000001
```

The server responds with nine bigendian unsigned integers that contain:

1. 36, the total length of the packet.
2. 32, the total length of the record.
3. 2, for SAD counters.

4. Supervisor state counter.
5. Problem state counter.
6. Wait state counter.
7. Wait for `manops` counter.
8. Emulate state (SIE) counter.
9. Supervisor state key 3 counter (active wait).

Programming Notes:

Notes:

1. The client program should be prepared for the counters to wrap. That is, if one sample contains a number smaller than the previous one, the delta is `0xffffffff-old+1+new`.
2. A counter can wrap at the earliest after 164 days.
3. Currently, `mgrsad` supports only one client at a time.

Utilities

The P390 utilities can be run by any user. The user must be authorised to access the files that the utility will open, if any.

awscke—3390 Disk Utility

The `awscke` utility performs operations on device media files:

- Create empty disks of any size up to a full model 3.
- Convert from the original device media file format (CKD_P370) or a *CMS Pipelines* dump of a volume to the extended format (CKE_P370). This reduces the size of the files significantly.
- Combine and convert old-format pinned files to CKE_P370 format.
- Combine CKE_P390 files into one. This is typically used to merge pinned files or a base device media file and pinned files.
- Reconcile the file contents and display statistics. Dump the contents of a particular cylinder and track.

The utility has a rather laid-back attitude to its operands; it quietly ignores operands that are not appropriate in any particular context.

awscke Flags

The table below describes all flags recognised by `awscke`.

Flg	Long name	Description
	-alternate	Dumps are formatted in the alternate format with both ASCII and EBCDIC.
	-ascii	Dumps that would normally be formatted for EBCDIC are formatted for ASCII instead.
	-batch	Number of tracks between progress messages when converting or verifying. The default is not to issue progress messages. If specified, the minimum value is 100.
	-block	Block number to dump for `-dumpfile`.
	-combine	Combine pinned files or base device media file and pinned files.
	-continue	Continue after the first error has been detected.
-c	-convert	Convert AWSCKD to AWSCKE. The input files may be zipped.
	-count	Number of input files. The default is 1.

Flg	Long name	Description
	-create	Create a new minidisk image. All tracks are initialised to zero as CPFMTXA would have done. By default, the volume label is the first six characters of the output file name in uppercase and all cylinders of the pack is allocated as PERM. See the flags `-label`, `-page`, and `-spool`. `-model` or `cylcount`.
	-cylcount	Number of cylinders for the new image file. The minimum is 1; the maximum is 3339.
	-cylinder	Cylinder number to dump. The default is cylinder 0.
	-datadir	Specify directory where disk image files are stored. The default is `/var/p390/drives`.
-d	-debug	Write debug messages.
	-dumpblock	Dump a 512-byte block of the file. Specify `-block` too.
	-dumpfile	Provide information on the file specified in the `-input` flag.
-s	-dumpstats	Display statistics.
	-dumptrack	Print contents of a track. Specify `-input`, `-cylinder`, and `-track`.
	-from	Specify the first pinned file to combine. The default is 1.
	-input	Input file to be converted or displayed. When `-count` is greater than 1, the string should include a "%d" to specify where the file sequence number is substituted. For `-trackwrite`, the file name may be a single hyphen, which specifies reading from standard input.
	-label	Specify the volume label for the CP volume being created. The string is converted to EBCDIC and uppercased. The default label is `-volume`, if specified, or the characters of the output file name up to the first period.
	-model	Model of 3390 to create. You can choose between a model 3.
	-nocompress	Do not compress output file. This is intended for debugging, but may also be specified when running on an extremely slow machine and you wish to trade disk space for performance.
	-oldfile	File name pattern for original device data file when used with `-verify`, but not `-convert`.
	-oldpinned	Convert old-format pinned files to CKE_P370 format.
	-output	Output file.

Flg	Long name	Description
	-page	Format the volume as a paging volume, except that cylinder 0 is PERM.
	-pincount	Specify the number of pinned files.
	-pinned	Specify pattern for the name of pinned files. The string `%d` is substituted with the file sequence number.
	-raw	Dump track without formatting. Used with `-dumptrack`.
	-readtracks	Read all tracks of the file to ensure that they are in the right place. That is, verify that the home address and record 0 contain the cylinder and head numbers that correspond to the track's number. This holds for all CP and CMS minidisks, but is not enforced by current hardware.
	-short	Dumps are formatted with 16 bytes per line.
	-spool	Format the volume as a spooling volume, except that cylinder 0 is PERM.
	-summary	Like `-dumpfile`, but display block usage statistics only.
	-timestamp	Put time stamp on progress messages.
	-track	Track number to dump. The default is track 0.
	-trackwrite	The input file is in the *trackwrite* format. You must specify the size of the volume by either `cylcount` or `model`.
-v	-verbose	Write high-level progress messages.
	-verify	Verify that the file created is identical to the original. You can specify this flag with `-convert`, `-oldpinned`, and `-trackwrite`; or by itself. When specified by itself, `-input`designates the CKE_P370 format file and `-oldfile`designates the original file, be that CKD_P370 or and old-format pinned file.
	-volume	Specify the volume label for an AWSCKE file. The file name constructed for the base file is `<datadir>/<volume>.cke` pinned files are named similarly. If `-pincount` is omitted, the number of pinned files is determined by looking for them. For `-create`, `-volume` also specifies the volume label.

The following table summarises which secondary flags are inspected for any particular function. Each cell contains:

Flag is ignored.
O Flag is optional.

R Flag is required.
r Flag is part of a set from which one must be selected.

Figure 1 (Page 1 of 2). awscke Function and Flag Cross Reference

Flag	combine	convert	create	dumpblock	dumpfile	dumptrack	findtrack	oldpinned	readtracks	summary	trackwrite	verify
batch	O	O						O	O		O	O
block				O	O					O		
continue												
count	O	O		O	O	O	O	O	O	O		
cylcount			r1								r	
cylinder						O	O					
datadir		O	O	O	O	O	O		O	O		
debug	O	O	O	O	O	O	O	O	O	O	O	O
dumpstats	O	O						O			O	
from	O											
input	r	r		r	r	r	r	R	r	r	R	R
label			O									
model			r1								r	
nocompress												
pinned	r											R
oldfile												R
output	R	R	r2					R			R	
page			O									
pincount	O	O		O	O	O	O		O	O		
raw					O	O				O		
spool			O									
summary					O							

Figure 1 (Page 2 of 2). awscke Function and Flag Cross Reference

Flag	**com-bine**	**con-vert**	**create**	**dump-block**	**dump-file**	**dump-track**	**find-track**	**old-pinned**	**read-tracks**	**sum-mary**	**track-write**	**verify**
timestamp	O	O	O	O	O	O	O	O	O	O	O	O
track						O	O					
verbose	O	O	O	O	O	O	O	O	O	O	O	O
verify		O						O				
volume	r	r	r2	r	r	r	r		r	r		

REXX Wrappers for awscke

There are several wrapper functions for the `awscke` utility. They all assume that you want the output files placed in the current directory.

awscvt—Convert CKD_P370 to CKE_P370

Specify the absolute or relative path to the first input file as the first word (that is, omit `-input`). After the input file, specify additional options you desire; you might consider `-verify` and `-batch 1000`.

The input file may be zipped or unzipped. It is assumed that the standard file naming conventions have been followed.

The output file name is of the form `<volid>.cke`.

awspin—Convert old Pinned Format to CKE_P370

Specify the input file name as the first word. It should be of the form `<prefix>_<sequence>.<devicenumber>`. The output file is of the form `<volID>_1.cke`, where the volume ID has been determined form the device number (123 or 124).

ckemerge—3390 Disk Utility

The `ckemerge` utility merges all pinned files for a volume into a single file.

- The present working directory must be the one in which the pinned files reside.
- `ckemerge` verifies that the pinned files are numbered sequentially starting with 1.
- The old pinned files are backed up by appending a tilde (~) to their names.
- The output file replaces the first pinned file.

Specify the volume label as the only argument.

Example:

```
cd /var/p390/drives
ckemerge 310res
```

Notes:

1. Be sure to change the pincount in the device map.
2. To merge a subset of the pinned files, use the `awscke` utility with suitable `-from` and `-pincount` arguments.

fmttrc—Format P390 Trace File

`fmttrc` formats the trace file written by the `manops` subcommand `dumptrace`. The format of the output from `fmttrc` is unspecified, but it should be readable ASCII.

The positional argument specifies the trace file to format. Unless `-stdin` or `-major` is specified, the default is `/var/p390/trace/dump`.

Flg	Long name	Description
-m	-major	Specify device major name for input trace file. The input file is `/var/%s/trace/dump`, where the device major replaces "%s". The default is `p390`.
-s	-stdin	Read input file from standard input. Ignore a positional argument.
-v	-verbose	Format timestamp information and other "header" information. The default is a more compact formatting.

sad—System Activity Display

`sad` displays a line showing the P390 system activity at a specified interval. `sad` requires no privileges to run.

`sad` can display its output on a single line that is continuously being overwritten; or it can write lines that will roll the screen in the normal way.

The line contains six segments showing the proportion of the interval spent in a particular state.

C	Colour	Description
s	red	Supervisor state, not key 3. CP overhead or Linux kernel.
p	green	Problem state.
w	blue	Wait state. The processor is idle or waiting for I/O.
h	yellow	Host wait time. The processor is waiting for `manops` to accept a request.
e	turquise	Emulator state. The processor is executing the Start Interpretive Execution (SIE) instruction, that is, a virtual machine is running.
a	purple	Supervisor state key 3. The processor is in "active wait", that is, CP has nothing to do and is polling for I/O interrupts.

Flg	Long name	Description
	-colours	A string that contains a number for each of the six colours in the order shown in the table above. The number is the RGB representation; it should be between 1 and 7. If you specify less than six characters, the balance remains unchanged.
-d	-debug	Increment the debug level.
	-host	The host name or dotted-decimal location of the server. The default is the local host.
	-interval	The interval length in seconds. The minimum interval is one second.

Flg	Long name	Description
	-nocolour	The line is written in black and white. Specify `-nocolour` when your terminal emulator does not process the escape sequences correctly.
	-overwrite	Overwriting the display on the previous. Thus, the screen remains static. Omit this flag to obtain a rolling display.
	-port	Specify the port to connect to. This is the port number that was specified when the server was started. Refer to “mgrsad—Server for System Activity Display” on page 42.
	-test	The server is not contacted; instead, an internal test case is run.
	-width	The number of characters in the display line. The minimum is 10. The default is 100.

verjtape—Verify the Integrity of a Tape Reel

The `verjtape` utility reads a tape reel file and verifies that the control information is intact. It does not verify whether the data records can be decompressed.

Specify the tape reel as the only argument.

Recovery Procedures

Dangling Device Managers

In some instances, it appears that `manops` “forgets” to terminate some device managers.

You can easily determine whether there are dangling device managers:

```
devmap.c: I/O subsystem ready.  17 highest subchannel.
end
Manager mgr3505 for device c ended exit status 0 11 active.
p390@Linux:  > ps
  PID TTY          TIME CMD
  774 pts/1    00:00:00 bash
  835 pts/2    00:00:00 bash
 1143 pts/1    00:07:38 mgr3215
 1144 pts/1    00:07:38 mgr3088c
 1145 pts/1    00:07:38 mgr3088c
 1507 pts/1    00:00:00 ps
```

The cure is to use the `kill` command to terminate these processes.

```
p390@Linux:  > kill 1143
p390@Linux:  > kill 1144
p390@Linux:  > kill 1145
p390@Linux:  > ps
```

Appendix A. Using Multiple P390 Cards in one System

The device driver code supports as many P390 cards as will fit on the PCI bus. P390 and P390E may be mixed.

You can connect the systems with channel-to-channel adapters and run then as an integrated system.

Device Major Names

The first card is associated with device major name `p390`, the second with `p391`, and so on.

Device Maps

`loadp390` expects a device map for each card. The second card's device map is `/home/p391/p391.devmap`.

Expanded storage

You can slice as many slices off RAM at boot time as you like by repeating the SLICE= boot parameter. The device manager will associate the largest slice with the first card (device major `p390`). Thus, if you have a P390E and a P390 and you want a larger slice for the P390E card, the p390E card must be scanned first.

Device Manager Data Files

You might wish to create `/var/p391` to hold the device manager data files for the second card.

You can share the disk image files as long as you use separate pinned files, but as it is unlikely that you will be running, say, two SFS servers, it is more likely that you will share only the base files.

Linux Users

To ensure that the two sets of device managers do not get into each others' hair, I recommend that you create a `p391` user and group for the second card and `chmod` subchannel files and device manager data files accordingly.

You may also consider making a `p390s` group to be the group for shared read-only disk image files that are used by all P390s.

Restrictions

These are the known restrictions when using multiple P390s in a system:

- You will need to modify the load procedure if you have more than ten cards in one system, as the eleventh card with have major device name `p400`.
- At most four network interfaces can be registered.
- Separate trace tables are allocated for each card. With many cards, this may be too large a drain on kernel resources.

Appendix B. Change Activity

4 Jul 2001

- Support for zipped input files.
- Introduce the `awscke` file format for 3390 data files and pinned files. The original format for pinned files is no longer supported. The `awscke` utility migrates files.
- For `mgr3390`, delete flags `loadpinned`, `closepinned`,
- The 3390 device manager now supports compressed tracks and performs space management in the pinned files.

27 Jul 2001

- Add mgr3423.
- Common commands are now scanned by the infrastructure in a separate command table. This table contains debug, nodebug, stop, system, superdebug.

26 Aug 2001

- Support for multiple P390 cards.
- Add support for up to four network interfaces per adapter card. This allows for additional VM stacks or a Linux/390 stack in a virtual machine.
- The index of the network interface is now appended automatically.
- The `mgr3088c` device manager is now documented. It was previously called `mgr3088`, which is to become a true CTC.

2 Sep 2001

While there are no changes to externals, the kernel module is now separated in one part that knows about the P390, but not about the kernel; and one part that knows about the kernel, but not the P390. The former part is OCO, while the latter part is supplied in source. Thus, the module should now be decoupled from the kernel release of the development system.

15 Sep 2001

- Add `-trackwrite` to the `awscke` utility. Also bring some of its documentation in accordance with reality.
- Rework patch for expanded storage. It is now specified by the `slice=` boot parameter.

30 Sep 2001

- Add `-combine` to the `awscke` utility.
- Fix problem with the CTC device manager for the network writing too large blocks for the VM stack.

13 Oct 2001

- Change to subchannel fragment as parameters for `startmgr` and `stopmgr` `manops` subcommands.
- Add `dumpfile` `manops` subcommand.
- Add `query` `manops` subcommand.

21 Oct 2001

- Add formatting options `label`, `page`, and `spool` for `awscke -create`.
- Add `starttask` and `stoptask` `manops` subcommands.

27 Oct 2001

- Add `mgrsad` started task. This is to be considered a pilot for a started task; the entire infrastructure is not yet in place.
- Add `sad` utility.
- Add `notimestamp` to `manops` and common device managers flags.
- Add LISTTASKs `manops` subcommand.

10 Nov 2001

- Add the flag `-major` to `manops`.
- The device map is renamed to <major>.devmap to allow for multiple device major names.
- Support up to 1G on P390E.
- Add ENV statement to the device map.
- Add environment variables P390_DIRECTORY, P390_TRACEFILE, P390_TRACEPATH, and P390_DEVICEPATH to control where device manager objects are stored in general.
- Add environment variable P390_DRIVES to specify where the 3390 device manager will find its disk image files.
- The path to the trace is now specified by the `-tracepath` argument; the `-tracefile` argument should specify only the file name pattern.
- Add documentation of the `verjtape` utility.
- Add the `ckemerge` utility.

16 Dec 2001

- Add `holdtime` and `maxhold` to `mgr3270`.
- Add `-major` to the trace format utility, `fmttrc`.
- Correct `manops` subcommand `dumptrace` to use the device major option specified, if any.

19 Jan 2002

- The 3423 device manager specifies the tape format with the `-format` string option rather than `-jphformat`. Thus, to take the default (and only currently supported format), specify `-format jphformat`.

9 Apr 2003

Fixed problems with the network interface to the CTC. The queue is now stopped unless we are ready to process packets. It is likely that there was a large overhead before this change. You can now `ifconfig` the interface down and up while the link is up and it will recover. The link will now come up when the P390 is started, even when IP traffic was attempted before the link was up.

Also reduces the amount of `syslog` noise when trying to present device ready DEs at startup.

27 Jun 2003

Stacked status returned on an 808 alert. Previously the 808 alert was ignored. This bites when the network interface presents attention before the device managers are running. No joy on that.

16 Oct 2004

- Fixed `awscke` to combine pinfiles correctly when the base file is not included.

6 Nov 2004

- Channel-to-channel adapter finally working over TCP/IP.

24 Mar 2005

- Fixed `mgr3390` to process the file mask and chaining requirements correctly for write count, key, data.
- Fixed `awscke` to convert from AWSCKD format. It is unclear when this broke.
- Add facility to trap subchannel activity in the device manager; and ability to run a device manager stand-alone using a file containing CCWs and operator commands.

26 Apr 2005

- Add 3370 device manager.
- Do not enforce a list of known device types for second word of DEVICE statement in the device map. The value has been verified and then ignored for quite some time.

30 Apr 2005

- Fix read backwards.
- Add support for named pipes in command input to `manops`.

5 May 2005

- Add `manops` commands ADD and MODIFY to add device managers dynamically and also change the program being run and its arguments.
- Generate CRW when a device manager is started for a device that was added after the P390 was started (that is, a device not specified in the device map).
- Print additional information on the QUERY command.

14 May 2005

- Add MODIFY DEVICE STOP command.
- Set return code for `manops` commands issued from a REXX program.
- Device managers no longer fail with EPIPE when the channel cancels a CCW due to a channel program check. As a consequence, it is now possible to IPL a virtual tape drive (the IPL simulator generates a channel program check).
- The 3390 device manager now returns correct sense on invalid CCW operation code. It processes non-multitrack Read Record 0 correctly when orientation is other than home address.

17 May 2005

- Add ID flag to `mgr3505`. Support a short card at the end of the deck. Present DE on `deck` and `rewind` commands.

25 May 2005

- Add ECHO `manops` command.
- Add `--liststats` flag for device managers. Add RESETSTATS and SHOWSTATS device manager commands.

9 Jun 2005

- Add D CARD, D MB, D WHOLECARD, and Q CPS `manops` commands.
- Fix `awscke`.

Index

Special Characters

A

D

E

F

I

K

L

M

N

P

R

S

T

U